CLOUD ROBOTICs: Vehículo autónomo conectado a AWS

Manuel Costanzo[1], Marcos Boggia[1], Ismael Rodríguez[1] and Armando De Giusti[1,2]

[1] *Instituto de Investigación en Informática LIDI (III-LIDI), Facultad de Informática, Universidad Nacional de La Plata.*

[2] *Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina.*

{mcostanzo, mboggia, ismael, degiusti}@lidi.info.unlp.edu.ar

Resumen

El trabajo desarrollado presenta un avance de la publicación realizada en el XXIII Congreso Argentino de Ciencias de la Computación (CACIC 2017), con título "Cloud Robotics: Auto Rover 4WD y Cuadricóptero controlados remotamente desde AWS" [1], en el cual se incorporan algoritmos para procesamiento y reconocimiento de imágenes, algoritmos para simulación y determinación de ruta óptima al destino establecido, algoritmos para detección de obstáculos a evitar y espacios transitables, con bajo costo computacional.

Se presenta el despliegue de un sistema multi-robot, compuesto por un chasis de auto Rover 4WD con cámara de video y sensores integrados, como así también de una cámara aérea fija (simulando la captura de imágenes desde un cuadricóptero no tripulado – Drone), conectados al Cloud público de Amazon Web Services (AWS). Asimismo, se detallan los prototipos desarrollados, el protocolo de comunicación utilizado con el servicio de AWS IoT, y los algoritmos implementados para efectuar el control remoto del vehículo de cuatro ruedas no tripulado, con el fin de llegar a un destino de forma autónoma.

Palabras claves: Autonomous Cars, Amazon Web Services, Cloud Robotics, Internet of Things, Protocolo MQTT, Herramienta de programación visual Node-RED.

1. Introducción

Los avances en el paradigma de Cloud Computing han provocado un factor disruptivo de las TI en la industria tecnológica. Según el Instituto Nacional de Estándares y Tecnologías del Departamento de Comercio de los EEUU (NIST), como en varias publicaciones de diversos autores, se ha definido a Cloud Computing como: "un paradigma informático de cómputo distribuido, que proporciona grandes conjuntos de recursos virtuales (como ser hardware, plataformas de desarrollo, almacenamiento y/o aplicaciones), fácilmente accesibles y utilizables por medio de una interfaz de administración web. Estos recursos son proporcionados como servicios (del inglés, "as a service") y pueden ser dinámicamente reconfigurados para adaptarse a una carga de trabajo variable (escalabilidad), logrando una mejor utilización y evitando el sobre o sub dimensionamiento (elasticidad). El acceso a los recursos se realiza bajo la demanda de los usuarios, en base a un modelo de autoservicio" [2].

El modelo de Cloud Computing presenta las características y beneficios siguientes: recursos disponibles bajo demanda, escalabilidad y elasticidad, aprovisionamiento automático de recursos y autoservicio.

Dicho paradigma, brinda al menos tres modelos de despliegue: Cloud Público, Cloud Privado y Cloud Híbrido [3,4].

Por otro lado, considerando la gran influencia de la robótica en la sociedad y la diversidad de servicios ofrecidos por robots, nos encontramos que los mismos presentan grandes limitaciones en consumo de energía, poder de cómputo, capacidad de almacenamiento, toma de decisiones, tareas cognitivas, etc.

En el año 2010, comenzaron a surgir proyectos de investigación (ej: RoboEarth [5]), que integraban las tecnologías de Cloud con los sistemas de robots. Es así, que James Kuffner propone el concepto de Cloud Robotics, basado en combinar las tecnologías de robots con el paradigma de Cloud Computing [6].

La idea de Cloud Robotics, permite por medio de aplicaciones tratar los datos de los componentes de hardware del robot (sensores, actuadores/motores, cámaras, microcontroladores, memoria, etc.), sin importar las limitaciones de cómputo de las placas de desarrollo con microcontroladores y la capacidad de almacenamiento de las mismas [7]. En otras palabras, este concepto permite a los robots obtener resultados de tareas de cómputo intensivo, tales

como: procesamiento de imágenes, reconocimiento de voz, determinación de rutas, confección de mapas, acciones cognitivas, etc., sin tratamiento local, sino en el Cloud.
Este paradigma brinda la capacidad de establecer escenarios para sistemas de multi-robot, donde cada robot se integra de un hardware mínimo; una placa microcontroladora con conectividad WiFi permite al robot comunicarse con el Cloud y otros robots. Los datos de sensores y/o sistemas de adquisición de imágenes se procesan en el Cloud y los actuadores de cada robot llevarán a cabo las operaciones necesarias [8].
El propósito de investigación del presente trabajo es el despliegue de un sistema de multi-robot, inicialmente con dos robots conectados al Cloud Público de Amazon Web Services con el fin de simular el desplazamiento autónomo de un vehículo de 4 ruedas, como así también la gestión de una cámara aérea (simulando la captura de imágenes desde un cuadricóptero no tripulado - Drone), ambos conectados al Cloud público de AWS, donde corre una instancia de EC2 con algoritmos de procesamiento de imágenes que determinan la ruta que debe efectuar el vehículo autónomo para llegar a un destino establecido. Para alcanzar tales objetivos, se determinó el método de comunicación con el Cloud; se registraron los robots como dispositivos o "cosas" en el servicio de AWS IoT [9]; también, se confeccionaron los prototipos de cada robot y se incorporaron los algoritmos para procesamiento y reconocimiento de imágenes, algoritmos para simulación y determinación de ruta óptima al destino establecido, algoritmos para detección de obstáculos a evitar y espacios transitables, con bajo costo computacional.
El presente trabajo está estructurado de la manera siguiente: en la Sección 2, se introducen algunos conceptos elementales. A continuación, la Sección 3, presenta los prototipos confeccionados y sus componentes. En la Sección 4, se describe el trabajo experimental realizado. Por último, la Sección 5, expone los resultados obtenidos en relación a esta investigación.

2. Conceptos básicos

2.1. Amazon Web Services

Es una plataforma de Cloud Público, que provee "Infraestructura como servicio"; a través de la tecnología de virtualización, provee un gran conjunto de recursos de cómputo, como ser, almacenamiento y capacidad de procesamiento, que pueden ser solicitados a demanda, como así también, se adecuan dinámicamente en tamaño conforme la necesidad del consumidor. Los servicios más destacados de AWS son "Elastic Compute Cloud" (EC2), "Simple Storage Service" (S3) y en nuestro caso utilizaremos los servicios de "AWS Internet of Things" (AWS IoT) [10].

2.2. Internet of Things (IoT)

Es un nuevo paradigma cuya definición deriva de considerar "objetos" o "cosas" conectadas a Internet por medio de redes Wireless, sean estas de tecnología WiFi o 4G LTE, utilizando protocolos de comunicación estándar. Dichas "cosas" pueden considerarse etiquetas de identificación por radio frecuencia (RFID), sensores, actuadores, teléfonos móviles, placas de desarrollos, etc. [11].

2.3. Protocolo MQTT

Message Queuing Telemetry Transport [12] es un protocolo de comunicación ligero, especialmente diseñado para tolerar conexiones intermitentes y reducir los requisitos de ancho de banda de la red. Desde finales del año 2014, fue presentado como un estándar abierto OASIS [13] y se ha convertido en el protocolo por excelencia para IoT. Soporta comunicación segura con TLS. Maneja tres niveles de calidad de servicio: QoS 0: A lo sumo una vez la entrega del mensaje. QoS 1: Al menos una vez la entrega del mensaje. QoS 2: Exactamente una vez la entrega del mensaje.
Este protocolo, implementa la comunicación de mensajes por medio de la publicación/suscripción sobre un tópico específico (canal de comunicación), como se puede observar en la figura siguiente:

Fig. 1 MQTT: publicación/subscripción de mensajes.

2.4. AWS IoT

Es un servicio que proporciona AWS con el fin de conectar, administrar y operar grandes conjuntos de dispositivos o cosas. Ofrece mecanismos de conectividad y seguridad para la transmisión de datos. También, permite que los datos, una vez enviados a la plataforma, puedan ser procesados por las aplicaciones de análisis masivo de datos (Elastic MapReduce), análisis predictivo para aprendizaje automático (Amazon Machine Learning), almacenamiento en bases de datos, etc. [14].
AWS IoT permite conectar fácilmente dispositivos al Cloud y a otros dispositivos. El servicio es compatible con los protocolos: HTTP, WebSockets y MQTT. Amazon ha optado por este último, por ende AWS IoT utiliza la especificación del protocolo MQTT v.3.1.1 con QoS 0 y 1.

Cada dispositivo debe ser registrado como una "cosa" en AWS IoT, para lo cual se emitirá un certificado y un par de llaves privada-pública para el mismo. El certificado y la llave privada, junto con el certificado de la Entidad Certificantes (CA) de Amazon, deberán ser instalados en el dispositivo con el fin que este pueda conectarse al servicio de AWS IoT previa autenticación necesaria.

2.5. Node-RED

Es una herramienta de programación visual, que permite programar algoritmos basados en flujos para IoT, sin la necesidad de escribir código. Provee de un editor de flujo basado en navegador WEB, que brinda una amplia paleta de nodos con diversa funcionalidad. El flujo se confecciona conectando los nodos entre sí. La funcionalidad de cada nodo está desarrollada sobre Node.js. Los flujos creados son almacenados utilizando JSON, lo cual facilita la importación o exportación para compartir los mismos [15].

3. Prototipos y componentes

Para el presente trabajo se ha ensamblado un chasis de robot Rover de cuatro ruedas, que simula un vehículo tradicional de propulsión trasera, es decir, con un motor que ejerce la transmisión del movimiento sobre el eje trasero; también, cuenta con la capacidad de orientar las ruedas delanteras para la dirección del mismo. Igualmente, se incorpora una cámara de video integrada para la detección de obstáculos de forma local al vehículo; y se utiliza una placa micro-controladora de bajo costo y un módulo de expansión compatible, para controlar el motor de la tracción trasera y un sensor de ultrasonido, como se detallan a continuación:

3.1. Chasis Rover 4WD con servo S3003

Es un chasis compuesto de dos capas de acrílico, con un motor de 3 a 24v DC y velocidades con carga de 258 a 347 rpm/min, cuatro ruedas y un servo S3003 que permite direccionar el chasis con giros de hasta 45 grados. Sus dimensiones son: 248mm (L) x 146mm (W) x 70mm (H).

Fig. 2 Chasis Rover 4WD.

3.2. Microcontroladora Raspberry Pi 3

Placa de bajo costo de desarrollo que brinda conexión WiFi, almacenamiento en memoria MicroSD de 32Gb y soporta alimentación de energía de 5v DC [16].

Fig. 3 Raspberry Pi 3.

3.3. RaspiRobot Board V3

Es un módulo de expansión [17] compatible con Raspberry Pi 3, que brinda la interfaz de conexión entre la placa de desarrollo y los motores; incluye una fuente de alimentación conmutada para el suministro de energía, tanto de la placa Raspberry Pi como de los actuadores.

Fig. 4 Raspirobot Board V3.

3.4. Sensor Ultrasónico HC-SR04

El HC-SR04 es un sensor de distancias por ultrasonidos capaz de detectar objetos y calcular la distancia a la que se encuentra en un rango de 2 a 450 cm; cuantifica el delay de la señal, es decir, el tiempo que demora la señal en regresar el eco.

Fig. 5 Sensor ultrasónico HC-SR04.

3.5. Raspberry Pi Camera Board

Pi Camera V2.1, es un módulo de video alta calidad de 8MP con sensor de imagen Sony IMX219 y un lente de foco fijo. Es capaz de tomar imágenes

estáticas de 3280 x 2464 pixeles como así también, capturar vídeo de 1080p30, 720p60 y 640x480p90.

Fig. 6 Raspberry Pi Camera Board V2.1.

3.6. Fuente conmutada Step-Down

Fuente de alimentacióncon regulador Step-Down DC-DC XL4015, que brinda una salida regulada de 1.25 a 36V.

Fig. 7 Fuente DC Step-Down.

3.7. Acelerómetro y Giroscopio de 3 ejes

EL módulo Acelerómetro MPU tiene un giroscopio de tres ejes con el que podemos medir velocidad angular y un acelerómetro también de 3 ejes con el que medimos los componentes X, Y y Z de la aceleración. Utilizamos el sensor de giroscopio para que el robot pueda posicionarse en la dirección correcta.

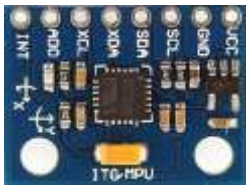

Fig. 8 Módulo MPU-6050.

4. Trabajo experimental

El trabajo de investigación consistió en desarrollar una plataforma web que permita controlar remotamente, vía internet, el sistema de multi-robot que emula un vehículo autónomo de 4 ruedas y una cámara de video de un Drone.
El escenario está compuesto por el Robot Rover 4WD, que posee una identificación por medio de la figura geométrica de un "cuadrado blanco con bordes negros". Así también, se dispone de un punto de destino identificado por la figura geométrica "circulo blanco con bordes negros" para la detección aérea y un "cuadrado blanco con bordes negros" para la detección desde el Robot Rover 4WD. Por otro lado, los objetos obstáculos están identificados por las figuras geométricas de "círculos rojos". Quien se encarga de identificar cada componente es el Cloud, por medio de las imágenes provenientes de la cámara aérea que emula a un Drone. Una vez identificados el Robot Rover 4WD y el punto destino sobre el escenario, se calcula el camino óptimo para que el vehículo se desplace hacia el punto desinto.
A continuación se describen los procedimientos para el procesamiento de imágenes y la determinación del camino óptimo.

4.1. Procesamiento de imágenes

El procesamiento de imágenes se realiza en una instancia de EC2 [18] en el Cloud, utilizando la librería OpenCV [19] y lenguaje de programación Python [20]. Se implementó un algoritmo que permite detectar e identificar sobre una imagen de captura aérea, las coordenadas de ubicación del Robot Rover 4WD con respecto al punto de destino; tomando como base dichas coordenadas se realiza el cálculo de la distancia entre el vehículo y el destino, como así también, los grados de diferencia entre los mismos.

4.1.1. Cámara aérea

La cámara aérea emula la captura de video que puede realizar un Drone. Dentro del escenario experimental se ubicó a una altura de tres metros sobre el nivel de la superficie del Robot Rover 4WD, enfocando el escenario completo. Dicha cámara está integrada a una placa de desarrollo Raspberry Pi 3.
Cuando el Robot aéreo recibe una señal vía el protocolo MQTT, desde el Cloud, solicitando una captura de imágen, el dispositivo realiza la captura de video en una resolución de 640x480 pixels, luego, disminuye la calidad en un 80% y realiza una compresión de la misma en base64, para por último enviarla vía MQTT a la instancia EC2, con el fin que el Cloud pueda realizar el reconocimiento y ubicación del Robot Rover 4WD y el punto de destino.
Todo el procesamiento mencionado se implementó con el lenguaje de programación visual Node-RED.

4.1.2. Cámara local

El Robot Rover 4WD, posee de forma integrada una cámara de video que permite realizar capturas de forma local al vehículo.
Cuando llega una notificación del Cloud, se efectua una captura de video y se realiza el procesamiento de compresión detallado anteriormente en la cámara aérea y se envía la imagen a la instancia EC2 en el Cloud.
A diferencia de la imágen capturada por la cámara aérea, en este caso, el punto destino es identificado por el reconocimiento de la figura geométrica "cuadrado blanco con bordes negros".

4.1.3. Instancia EC2

Cuando la instancia EC2, recibe una imagen, vía MQTT, se procede a identificar las figuras geométricas en la imágen. En el caso de la imagen capturada por la cámara aérea, como ya se ha mencionado, se tiene que detectar un "cuadrado blanco con bordes negros" para identificar al Robot Rover 4WD; y la figura geométrica de un "círculo blanco con bordes negros" para identificar al punto de destino. Por otro lado, en el caso de la imagen capturada por la cámara local del Robot Rover 4WD, se debe identificar la figura geométrica de un "cuadrado blanco con bordes negros" para identificar al punto de destino.
Una vez detectadas las figuras, se prosigue a calcular la distancia entre el vehículo y el destino. Previamente a este cálculo, es necesario haber calibrado las cámaras. Debido a que la visión de las cámaras es distinta, ya que la cámara aérea puede observar el escenario completo, en cambio, la cámara local sólo el punto de destino, la calibración de las mismas también difiere.
La figura 9 presenta el flujo de procesamiento de una imagen.

Fig. 9 Flujo de procesamiento de imágenes.

4.1.4. Calibración cámara aérea

El proceso de calibración se basa en realizar cálculo de la medida del ancho (en centímetros) de la figura geométrica del "cuadrado blanco con bordes negros" del Robot Rover 4WD. El objetivo de la calibración es determinar cuántos "centímetros" hay en un pixel. Con esta referencia, se podrá conocer la distancia en centímetros entre cualquier par de pixels.
Para lograr tal calibración, se calcula la distancia Euclidiana [21] entre dos extremos de la figura geométrica del cuadrado, y se divide por el ancho de referencia, como se puede observar en el algoritmo 1.

Algoritmo 1 Distancia Euclidiana

```
1: # Librería para calcular la distancia
   euclidiana en PyTHON
2: from scipy.spatial import distance as dist
3: def get_distance(p1, p2, reference):
4:   return dist.euclidean(p1, p2) / reference
```

Ejecutado el algoritmo 1, se logra obtener el conocimiento de cuántos pixels forman un "centímetro".
Dicha información es útil para la determinación de la distancia en "centímetros" entre el Robot Rover 4WD y el punto de destino; ahora alcanza con realizar el cálculo las coordenadas (x,y) del centro de las figuras geométricas, e invocar a la función del algoritmo 1, reemplazando los parámetros p1, p2 y reference por las coordenadas del Robot Rover 4WD, las del punto destino y el resultado de la calibración respectivamente.

4.1.5. Calibración cámara local

Este proceso de calibración [22] difiere del anteriormente mencionado, ya que no calcula la distancia entre objetos sobre una imágen, por el contrario, requiere conocer la distancia entre la cámara local y la figura geométrica "cuadrado blanco con bordes negros" del punto destino. Para esta calibración es necesario contar con dos datos: el ancho de la figura geométrica y la distancia en que se encuentra el Robot Rover 4WD con respecto de la figura al momento de la calibración; con dichos datos, se procede a realizar el cálculo de la longitud focal de la cámara local por medio de la ecuación 1:

$$\text{Longitud\_Focal} = \frac{WFigPX * Distancia}{WenCM} \quad \text{(ecuación 1)}$$

Donde:

WFigPX: es el ancho de figura en Pixels
WenCM: Ancho en centímetros.

El cálculo de la distancia focal se obtiene tras la ejecución del algoritmo 2.

Algoritmo 2 Longitud Focal

```
1: # Librería de Opencv
2: import cv2
3: def focal_length(contour, width, distance):
4:   marker = cv2.minAreaRect(contour)
5:   return (marker[1][0] * distance) / width
6: #marker[1][0] retorna el ancho en pixels
   de la figura geométrica.
```

Una vez obtenida la distancia focal, se procede a calcular la distancia entre la cámara y cualquier

objeto en la imagen local, por medio de la ecuación 2:

$$\text{Dist\_Objeto} = \frac{WenCM * Longitud\_Focal}{WFigPX} \text{ (ecuación 2)}$$

Donde:

WenCM: Ancho en centímetros.
WFigPX: es el ancho de figura en Pixels

La distancia al objeto se obtiene ejecutando el algoritmo 3.

Algoritmo 3 Distancia Objeto

```
1:  # Librería de Opencv
2:  import cv2
3:  Def distance_from_object_to_camera(contour, width, focal_length):
4:    marker = cv2.minAreaRect(contour)
5:    return (width * focal_length) / marker[1][0]
```

4.2. Planificación de camino óptimo

Una vez finalizada el procesamiento de las imágenes y realizado el cálculo de distancia entre el Robot Rover 4WD y el punto de destino, se procede a determinar el camino óptimo.

Un camino está compuesto por un conjunto de movimientos que guiarán al vehículo autónomo desde la posición donde se encuentra, hacia el punto de destino, evitando que colisione con los obstáculos que se presenten de por medio. Es necesario aclarar que si bien el algoritmo de planificación, que se detalla a continuación, contempla la identificación y el rodeo de los obstáculos, este trabajo se limita a que el Robot Rover 4WD alcance el punto de destino sin obstáculos de por medio.

El Robot Rover 4WD puede realizar movimientos de giro en grados a derecha e izquierda (de 0 a 180 grados y de 0 a -180, respectivamente), como así también, movimientos para avanzar una determinada distancia en centímetros.

La técnica de planificación elegida es la denominada RRT (Rapidly Exploring Random Trees) [23]. Este algoritmo se basa en la construcción de un árbol de configuraciones que crece buscando a partir de un punto origen en forma aleatoria, como se puede observar en el algoritmo 4.

Previamente, se definen algunas variables:

- self.node_list: es el árbol generado
- self.start, self.end: son las coordenadas de inicio y de fin respectivamente
- self.rand_y, self.rand_x: es el espacio de configuraciones
- self.expand_dis: es la distancia en el que los nodos se expanden. En nuestro caso utilizamos la distancia euclidiana calculada en la calibración de la cámara aérea, por lo que los nodos están separados a un centímetro de distancia.

Algoritmo 4 Algoritmo RRT

```
1:  def planning(self):
2:      # Flag que indica si encontró el destino.
3:      goal_flag = False
4:      # Lista de nodos del path.
5:      self.node_list = [self.start]
6:    for i in range(MAX_ITER):
7:        # Búsqueda aleatoria
8:        if random.randint(0, 100) > self.goal_sample_rate:
9:            rnd = [random.uniform(self.rand_x[0], self.rand_x[1]), random.uniform(
10:               self.rand_y[0], self.rand_y[1])]
11:       else:
12:           rnd = [self.end.x, self.end.y]
13:       # Busca el nodo más cercano.
14:       nind = self.get_nearest_list_index(self.node_list, rnd)
15:       nearest_node = self.node_list[nind]
16:       theta = math.atan2(rnd[1] - nearest_node.y, rnd[0] - nearest_node.x)
17:       new_node = copy.deepcopy(nearest_node)
18:       new_node.x += self.expand_dis * math.cos(theta)
19:       new_node.y += self.expand_dis * math.sin(theta)
20:       new_node.parent = nind
21:       if not self.collision_check(new_node, self.path.obstacles):
22:           continue
23:       self.node_list.append(new_node)
24:       # Verifica si se llegó al destino.
25:       dx = new_node.x - self.end.x
26:       dy = new_node.y - self.end.y
27:       d = math.sqrt(dx**2 + dy**2)
28:       if d <= self.expand_dis:
29:           goal_flag = True
30:           print("Goal!!")
31:           break
```

En la línea 9, se inserta el nodo inicial en la estructura del árbol de nodos.

En la línea 10, se declara el límite de repeticiones del algoritmo. Este límite está dado por el valor que se le da a la constante MAX_ITER.

Entre las líneas 13 y 18 se realiza la búsqueda aleatoria del nodo dentro del espacio de configuraciones.

En la línea 22, se obtiene el vecino más próximo de

la configuración aleatoria obtenida anteriormente.
Entre las líneas 21 y 23, se determina el nuevo nodo a agregar al árbol realizando un salto (de tamaño self.expand_dis * math.cos(theta)) partiendo del vecino más cercano y el nodo aleatorio. Si se determina, que en el camino no hay una colisión, se agrega el nuevo nodo al árbol. Por el contrario, si efectivamente hay un obstáculo de por medio, se continúa con la iteración siguiente y no se agrega el nodo al árbol resultante.
Entre las líneas 24 y 31, se verifica si se llegó al punto destino. Si la distancia entre el nuevo nodo y el punto destino es menor a la métrica establecida (un centímetro), se termina el procesamiento, obteniendo el árbol de nodos. Si la distancia no es menor, se continúa con la iteración siguiente.

Se implementó una interfaz gráfica utilizando node-red-dashboard [24], que permite visualizar el resultado de la ejecución de la simulación del algoritmo RRT.
Se puede observar en la figura 10, el resultado de simular la ejecución del algoritmo RRT tomando como punto de inicio la ubicación del Robot Rover 4WD (punto verde), el punto de destino (punto azul) y los obstáculos de por medio (puntos rojos).

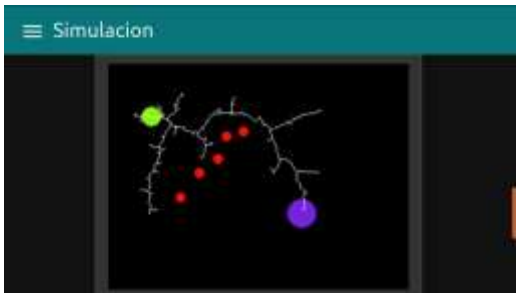

Fig. 10 Simulación de la ejecución del algoritmo RRT.

Como resultado de la planificación se obtiene un vector, donde cada una de sus celdas posee una coordenada (x,y). Se debe transformar estas coordenadas en movimientos. Cada movimiento constará de un grado de giro y una distancia en centímetros. Para dicha transformación se utilizar el algoritmo 5.

Algoritmo 5 Transformación a movimiento

```
1:  def load_movements(self, prev_deg):
2:      p1 = None
3:      for p2 in coords:
4:          if p1:
5:              distance = helper.get_distance(p1,
        p2, self.config.pixel_distance)
6:              xDiff = p2[0] - p1[0]
7:              yDiff = p2[1] - p1[1]
8:              degree = 360 -
        (math.degrees(math.atan2(yDiff,
        xDiff))%360)
9:              degree_difference = ((degree -
        prev_deg) + 180) % 360 - 180
10:             prev_deg = degree
11:   self.movements.append(degree_difference,
        distance])
12:         p1 = p2
```

En la Línea 1, se declara el encabezado de la función, que consta de un parámetro llamado "prev_degree". Este es el grado inicial del Robot Rover 4WD; a partir de este grado se podrá ir calculando la diferencia de grado entre los demás puntos.
En la línea 3, comienza el bucle, iterando por todas las coordenadas resultantes de la planificación del camino.
En la línea 4, se calcula la distancia Euclidiana ya mencionada anteriormente.
En las líneas 6 y 7, se calculan las diferencias entre x2-x1, y2-y1.
En las líneas 8 y 9, se calculan la diferencia de grados entre el punto anterior y un nuevo punto. Los grados varían en 0, 180 y 0, -180.
Por último, en la línea 12 se almacena el movimiento.

4.3. Plataforma WEB

Se desarrolló una plataforma WEB que brinda una interfaz gráfica; esta permite realizar la calibración de las cámaras como así también, visualizar el resultado de la ejecución del algoritmo de planificación de caminos RRT simulado. Por otro lado, permite dar inicio al sistema de multi-robots, visualizando en tiempo real todas las acciones que se estén llevando a cabo con el Robot Rover 4WD, los resultados de la planificación de la trayectoria gráficamente y las imágenes capturadas, tanto del vehículo autónoma como de la cámara aérea.

4.3.1. Calibración

Esta sección de la plataforma WEB, permite al usuario el ingreso de la información necesaria para calibrar las cámaras, como ser: tipo de figura geométrica ("cuadrado" o "circulo"), medida de ancho de la figura, etc.. Además, permite efectuar la calibración tanto del Robot Rover 4WD y la cámara aérea (Drone) notificando con un mensaje de éxito o de error.

Fig. 11 Interfaz de Calibración.

4.3.2. Simulación

Esta sección de la plataforma WEB, permite al usuario simular la planificación de un camino, permitiendo ingresar un punto de inicio, un punto de destino y diversos puntos de obstáculos, dando a elegir el radio y color de los mismos, junto con opciones para eliminar los obstáculos y para resetear la simulación. Además, permite la posibilidad de visualizar el resultado final o la animación de cómo el algoritmo de planificación RRT fue creando el árbol de nodos.

Fig. 12 Interfaz de Simulación sin animación.

Fig. 13 Interfaz de Simulación con animación.

4.3.3. Tiempo Real (Live)

Esta última sección de la plataforma WEB, permite dar inicio y fin al sistema Multi-Robots, y visualizar en tiempo real qué información se está intercambiando y en qué situación está cada actuador. Se puede apreciar gráficamente el resultado de la planificación del camino, y ver las imágenes que las cámaras están enviando a la instancia EC2 en el Cloud

Fig. 14 Interfaz de gestión del sistema Multi-Robots.

5. Resultados obtenidos

En esta investigación se ha implementado un sistema multi-robot conectado al Cloud público de AWS. Se han confeccionado los prototipos para emular un vehículo autónomo de 4 ruedas y un dispositivo de vuelo ambos no tripulados. Se han desarrollado los algoritmos que permiten realizar procesamiento de imágenes, reconocimiento de figuras geométricas, determinación del camino óptimo.

Además, se implementó una plataforma WEB para poder gestionar el sistema multi-robot como así también, poder visualizar el resultado de simular la ejecución del algoritmo de planificación de camino óptimo RRT.

Uno de los avances más significativos de esta investigación consiste en que, actualmente, el vehículo de cuatro ruedas conectado a AWS, se puede orientar, desplazar y arribar al destino establecido, de forma autónoma.

Podemos concluir y reafirmar que Cloud Robotics es una tecnología que favorecerá el avance de los sistemas multi-robots reduciendo las limitaciones que se presentan actualmente.

Referencias

[1] A. De Giusti, I. Rodríguez, M. Costanzo and M. Boggia, “Cloud Robotics: Auto Rover 4WD y Cuadricóptero controlados remotamente desde AWS” in CACIC ’17. *Proceedings of XVIII Workshop de Procesamiento Distribuido y Paralelo (WPDP) – XXIII Congreso Argentino de Ciencias de la Computación. La Plata, Argentina*. 2017.

[2] P. Mell and T. Grance, “The NIST Definition of Cloud Computing”. Publicación Especial 800-145. Septiembre, 2011.

[3] J.E. Pettoruti, I. Rodriguez, F. Chichizola, A. De Giusti. “Análisis de la degradación de las comunicaciones en algoritmos de cómputo científico en un Cloud privado”. In CACIC ’12. *Proceedings of XII Workshop de Procesamiento Distribuido y Paralelo (WPDP) – XVIII*

Congreso Argentino de Ciencias de la Computación. Bahía Blanca, Argentina, 2012.

[4] I. Rodriguez, J.E. Pettoruti, F. Chichizola and A. De Giusti: "Despliegue de un Cloud Privado para entornos de cómputo científico". In CACIC '11. *Proceedings of XI Workshop de Procesamiento Distribuido y Paralelo (WPDP) - XVII Congreso Argentino de Ciencias de la Computación. La Plata, Argentina*. 2011.

[5] "RoboEarth" Available at: http://www.roboearth.org. Accessed on 2018-06-06.

[6] J. Kuffner. "Cloud-enabled robots". *IEEE-RAS International Conference on Humanoid Robot. Nashville, USA*, 2010.

[7] L. Wang, M. Liu, M. Meng, R. Siegwart. "Towards Real-Time Multi-Sensor Information Retrieval in Cloud Robotic System". *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI). Hamburgo, Alemania*, 2012.

[8] L. Turnbull: "Cloud Robotics: Formation Control of a Multi Robot System Utilizing Cloud Infrastructure". *Proceedings of IEEE – Southeastcon. Jacksonville, USA*, 2013.

[9] "AWS IoT" Available at: https://aws.amazon.com/es/iot. Accessed on 2018-06-06.

[10] I. Bermudez, S. Traverso, M. Mellia and M. Munafò. "Exploring the cloud from passive measurements: The Amazon AWS case". *Proceedings of IEEE – INFOCOM*, 2013.

[11] L. Atzori, A. Iera and G. Morabito. "The Internet of Things: A survey". *Journal Computer Networks ELSEVIER*. Volume 54, Issue 15. 2010.

[12] "Protocolo MQTT" Available at: http://mqtt.org. Accessed on 2018-06-06.

[13] "OASIS MQTT", Available at: https://www.oasis-open.org. Accessed on 2018-06-06.

[14] "AWS IoT" Available at: https://aws.amazon.com/es/iot/. Accessed on 2018-06-06.

[15] "Node-RED" Available at: https://nodered.org. Accessed on 2018-06-06.

[16] "The official site RapsberryPi" Available at: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. Accessed on 2018-06-06.

[17] "Raspirobot Board V3" Available at: https://www.monkmakes.com/rrb3/. Accessed on 2018-06-06.

[18] "Tipos de instancias de Amazon EC2" Available at: https://aws.amazon.com/es/ec2/instance-types/. Accessed on 2018-06-06.

[19] "Open Source Computer Vision Library" Available at: https://opencv.org/. Accessed 2018-06-06.

[20] "Python Software Fundation" Available at: https://www.python.org/. Accessed on 2018-06-06.

[21] J.J. Esqueda Elizondo and L.E. Palafox Maestre, *Fundamentos de procesamiento de imágenes*. Baja California, México: Universidad Autónoma de Baja California, 2005.

[22] A. Hernandez Zavala and J.A. Huerta Ruelas, "Sistema de medición de distancia mediante imágenes para determinar la posición de una esfera utilizando el sensor Kinect XBOX", *Journal of Polibits*, vol 49, 2014.

[23] D.A. López García, *Nuevas aportaciones en algoritmos de planificación para la ejecución de maniobras en robots autónomos no holonomos*. PhD thesis, Universidad de Huelva, 2011.

[24] "Node-Red-Dashboard" Available at: https://github.com/node-red/node-red-dashboard. Accessed on 2018-06-06.